UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/612,583	07/01/2003	Ronald P. Doyle	RSW9-2003-0069US1 (7161-9	6219

46320 7590 09/10/2007

CAREY, RODRIGUEZ, GREENBERG & PAUL, LLP
STEVEN.M. GREENBERG
950 PENINSULA CORPORATE CIRCLE
SUITE 3020
BOCA RATON, FL 33487

EXAMINER	
MEHRMANESH, ELMIRA	

ART UNIT	PAPER NUMBER
2113	

MAIL DATE	DELIVERY MODE
09/10/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/612,583
Filing Date: July 1, 2003
Appellant(s): Ronald DOYLE, et al.

Scott D. Paul
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed June 1, 2007 appealing from the Office action mailed March 7, 2007.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments

The appellant's statement of the status of amendments contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Cobb et al. (U.S. Patent No. 5,119,377)

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-15 are rejected under 35 U.S.C. 102(b) as being anticipated by Cobb et al. (U.S. Patent No. 5,119,377).

As per claim 1, Cobb discloses a method for autonomically diagnosing and correcting error conditions in a computing system (col. 2, lines 48-52) of interrelated components and resources (Fig. 10), the method comprising the steps:

For each one of the components, reporting error conditions in a log file (col. 4, lines 21-29) using both uniform conventions for naming dependent ones of the interrelated components and resources (col. 4, lines 26-29) and (Fig. 1, *application data table*) and also a common error reporting format (col. 5, lines 57-68 through col. 6, lines 1-8). Cobb discloses the EDDC process uses the application data table information to generate a dump of specific program storage areas, to create an entry in a software error log (col. 4, lines 26-29). He also discloses using a sequence-naming convention (col. 5, lines 67-68), which is a common error-reporting format.

Detecting error conditions (col. 3, lines 55-59) arising from individual ones (col. 3, lines 49-50) of the interrelated components (Fig. 7) and (col. 4, lines 21-29)

Responsive to detecting an error condition in a specific one of the components (col. 4, lines 44-50), parsing a log associated with said specific one of the components (col. 6, lines 36-39) to determine whether said error condition arose from a fault in one of the interrelated components and resources named in said associated log (col. 4, lines 51-61)

And further parsing a log associated with said one of the interrelated components and resources to identify a cause for said fault (col. 4, lines 62-68 through col. 5, lines 1-11) and (col. 10, lines 11-13, 40-41) and correcting said fault (col. 5, lines 12-20) and (col. 10, lines 67-68).

As per claim 2, Cobb discloses inserting analysis code (col. 5, lines 32-37, *error detection code*) in said specific one of the components (col. 6, lines 36-39) responsive to detecting said error condition (col. 5, lines 32-37) said analysis-code having a configuration for reporting operational data associated with said error condition (col. 9, lines 58-64), and utilizing said reported operational data to identify a cause for said error condition (col. 10, lines 11-13, 40-41).

As per claim 3, Cobb discloses activating dormant analysis code (col. 5, lines 32-37, *error detection code*) in said specific one of the components (col. 6, lines 36-39) responsive to detecting said error condition (col. 5, lines 32-37) said dormant analysis code having a configuration for reporting operational data associated with said error condition (col. 9, lines 58-64), and utilizing said reported operational data to identify a

cause for said error condition (col. 10, lines 11-13, 40-41).

As per claim 4, Cobb discloses inserting analysis code (col. 5, lines 32-37, *error detection code*) in both said specific one of the components (col. 6, lines 36-39) and said one of the interrelated components and resources responsive to detecting said error condition (col. 5, lines 32-37), said analysis code having a configuration for reporting operational data for said specific one of the components and said one of the interrelated components and resources (col. 9, lines 58-64)

utilizing said reported operational data to correlate error conditions in each of said specific one of the components and said one of the interrelated components and resources to identify a cause for said error condition (col. 10, lines 11-13, 40-41).

As per claim 5, Cobb discloses inserting analysis code (col. 5, lines 32-37, *error detection code*) in said specific one of the components (col. 6, lines 36-39) responsive to detecting said error condition (col. 5, lines 32-37) said analysis code having a configuration for suspending the operation of said specific one of the components pending resolution of said error condition (col. 2, lines 67-68 through col. 3, lines 1-2).

As per claim 6, Cobb discloses correcting step comprises the steps of:

determining from said further parsing step whether said fault in said one of the interrelated components and resources named in said associated log arose from an additional fault in yet another one of the interrelated components and resources (col. 6,

lines 50-61) and, repeating each of the parsing and correcting steps for said yet another interrelated one the components and resources (col. 4, lines 62-68 through col. 5, lines 1-11) and (col. 10, lines 11-13, 40-41) and correcting said fault (col. 5, lines 12-20) and (col. 10, lines 67-68).

As per claim 7, Cobb discloses an autonomic system for diagnosing and correcting error conditions (col. 2, lines 48-52) among interrelated components and resources (Fig. 10) comprising:

A plurality of commonly formatted log files (col. 4, lines 21-29) utilizing standardized naming conventions for the interrelated components and resources (col. 4, lines 26-29) and (Fig. 1, *application data table*), each of said commonly formatted log files having an association with one of the interrelated components and resources (col. 5, lines 57-68 through col. 6, lines 1-8). Cobb discloses the EDDC process uses the application data table information to generate a dump of specific program storage areas, to create an entry in a software error log (col. 4, lines 26-29). He also discloses using a sequence-naming convention (col. 5, lines 67-68), which is a common error-reporting format.

an autonomic system administrator (Fig. 10) coupled to each of the interrelated components and resources (col. 6, lines 36-39) and configured to parse said log files to identify both error conditions arising in associated ones of the interrelated components and resources (Fig. 10), and also dependent ones of the interrelated components and resources giving rise to the identified error conditions (col. 4, lines 62-68 through col. 5,

lines 1-11) and (col. 10, lines 11-13, 40-41).

As per claim 8, Cobb discloses a codebase of analysis code (Fig. 1) and, code insertion logic (col. 5, lines 32-37, *error detection code*) coupled to said autonomic system administrator and programmed to insert portions of said analysis code in selected ones of the interrelated components and resources (col. 6, lines 36-39).

As per claim 9, Cobb discloses analysis code comprises byte code and wherein said code insertion logic comprises byte code insertion logic (col. 10, lines 14-20).

As per claim 10, Cobb discloses a machine readable storage having stored thereon a computer program (Fig. 10) for autonomically diagnosing and correcting error conditions in a computing system (col. 2, lines 48-52) of interrelated components and resources (Fig. 10), the computer program comprising a routine set of instructions for causing the machine to perform the steps:

For each one of the components, reporting error conditions in a log file (col. 4, lines 21-29) using both uniform conventions for naming dependent ones of the interrelated components and resources (col. 4, lines 26-29) and (Fig. 1, *application data table*) and also a common error reporting format (col. 5, lines 57-68 through col. 6, lines 1-8). Cobb discloses the EDDC process uses the application data table information to generate a dump of specific program storage areas, to create an entry in a software

error log (col. 4, lines 26-29). He also discloses using a sequence-naming convention (col. 5, lines 67-68), which is a common error-reporting format.

Detecting error conditions (col. 3, lines 55-59) arising from individual ones (col. 3, lines 49-50) of the interrelated components (Fig. 7) and (col. 4, lines 21-29)

Responsive to detecting an error condition in a specific one of the components (col. 4, lines 44-50), parsing a log associated with said specific one of the components (col. 6, lines 36-39) to determine whether said error condition arose from a fault in one of the interrelated components and resources named in said associated log (col. 4, lines 51-61)

And further parsing a log associated with said one of the interrelated components and resources to identify a cause for said fault (col. 4, lines 62-68 through col. 5, lines 1-11) and (col. 10, lines 11-13, 40-41) and correcting said fault (col. 5, lines 12-20) and (col. 10, lines 67-68).

As per claim 11, Cobb discloses inserting analysis code (col. 5, lines 32-37, *error detection code*) in said specific one of the components (col. 6, lines 36-39) responsive to detecting said error condition (col. 5, lines 32-37) said analysis-code having a configuration for reporting operational data associated with said error condition (col. 9, lines 58-64), and utilizing said reported operational data to identify a cause for said error condition (col. 10, lines 11-13, 40-41).

As per claim 12, Cobb discloses activating dormant analysis code (col. 5, lines 32-37, *error detection code*) in said specific one of the components (col. 6, lines 36-39) responsive to detecting said error condition (col. 5, lines 32-37) said dormant analysis code having a configuration for reporting operational data associated with said error condition (col. 9, lines 58-64), and utilizing said reported operational data to identify a cause for said error condition (col. 10, lines 11-13, 40-41).

As per claim 13, Cobb discloses inserting analysis code (col. 5, lines 32-37, *error detection code*) in both said specific one of the components (col. 6, lines 36-39) and said one of the interrelated components and resources responsive to detecting said error condition (col. 5, lines 32-37), said analysis code having a configuration for reporting operational data for said specific one of the components and said one of the interrelated components and resources (col. 9, lines 58-64)

utilizing said reported operational data to correlate error conditions in each of said specific one of the components and said one of the interrelated components and resources to identify a cause for said error condition (col. 10, lines 11-13, 40-41).

As per claim 14, Cobb discloses inserting analysis code (col. 5, lines 32-37, *error detection code*) in said specific one of the components (col. 6, lines 36-39) responsive to detecting said error condition (col. 5, lines 32-37) said analysis code having a configuration for suspending the operation of said specific one of the components

pending resolution of said error condition (col. 2, lines 67-68 through col. 3, lines 1-2).

As per claim 15, Cobb discloses correcting step comprises the steps of: determining from said further parsing step whether said fault in said one of the interrelated components and resources named in said associated log arose from an additional fault in yet another one of the interrelated components and resources (col. 6, lines 50-61) and, repeating each of the parsing and correcting steps for said yet another interrelated one the components and resources (col. 4, lines 62-68 through col. 5, lines 1-11) and (col. 10, lines 11-13, 40-41) and correcting said fault (col. 5, lines 12-20) and (col. 10, lines 67-68).

(10) Response to Argument

In response to appellant's arguments regarding independent claims 1, 7, and 10, that Cobb fails to disclose "interrelated components and resources", "using uniform conventions for naming dependent ones of the interrelated components", "detecting error conditions arising from individual ones of the interrelated components", and "parsing a log associated with said specific one of the components to determine whether said error condition arose from a fault in one of the interrelated components and resources named in said associated log", the Examiner respectfully disagrees for the following reasons:

Interrelated components and resources

Appellant's argues that Cobb fails to disclose interrelated components and resources. The Examiner respectfully disagrees and would like to point out to the sections of the applicant's specification, which have been reproduced below for clarification purposes.

The interrelated components and resources as disclosed by the applicant include, for instance, *application components, operating system components, and database manager.* The specification (see specification, page 10, paragraph [0019], lines 1-6) states, "the system resources 140 can range from a database manager which regulates access to a database management system, to a communications controller and is not limited to any specific computing resource. By comparison, the system components 130 can range from application components, including Web services, to operating system components and services."

The specification further provides examples of the different error categories for the components and resources (see specification, page 11, paragraph [0021], lines 5-8), wherein the applicant discloses "...such as would be the case where *invalid data input* has been provided to the system component 130, or whether the error condition has arisen based upon the *unexpected behavior* of a dependent one of the system components 130 or a dependent one of the resources 140."

Cobb discloses examples of the different error categories for the interrelated components and resources, including invalid data, and unexpected behavior of the components and resources (see col. 5, lines 50-54, wherein Cobb discloses "If a

software routine is called or requested to perform a service from another software routine (e.g., get storage, read a database) and the call was found to be in error (e.g., the **function is not supported**, **invalid data** in the call)").

In light of the specification and based on the disclosed error category examples of Cobb (i.e. a software routine calling/requesting to perform a service from another software routine), Cobb's computer system implicitly comprises interrelated components and resources (i.e. database management system, application components), which reads on the claimed limitations as recited in claims 1, 7, and 10.

Using uniform conventions for naming dependent ones of the interrelated components and resources

Appellant's argues that Cobb fails to disclose uniform conventions for naming dependent ones of the interrelated components and resources. The Examiner respectfully disagrees and would like to point out to Cobb's Figs. 8A and 8B, which show the error log format.

With respect to the uniform conventions, the applicant's specifications (see specification, page 10, paragraph [0020], lines 4-6) states that "Each entry can be formatted according to a common error format in which all error conditions, regardless of source or nature, are expressed **uniformly in a standardized way**."

Cobb discloses the error detection keywords as shown in TABLE 1 (col. 6, lines 62-68 through col. 7, lines 1-37). Noting col. 6, lines 55-58, wherein Cobb discloses "Each symptom string entity has a format `xxxx/vvvvvvvv` where `xxxx` is a keyword

that identifies the category of the associated value, `vvvvvvvv`". The format of the entries of the error log as shown in sections 1-4 of Fig. 8A and 8B is expressed uniformly in a standard way.

In addition the above keywords as disclosed by Cobb (col. 7, lines 1-37), are used to not only identify errors in the components/resources, but also their associated components/resources (i.e. data structure, device, register, subroutine, function or procedure name associated with the failure detected). The following are some examples of these keywords as shown in TABLE 1:

`DEVS`--specifies that `vvvvvvvv` contains **a value that identifies a device that is involved in a failure** (e.g., device number, device address).

`FLDS`--specifies that `vvvvvvvv` contains **a field, data structure, or label name associated with the error detected**.

`MS`--specifies that `vvvvvvvv` contains **a message number, issued by a program or device, that is associated with the error detected**.

`RIDS`--specifies that `vvvvvvvv` contains a module, macro, **subroutine, function or procedure name associated with the failure detected.**

In light of the specification and based on the disclosed error format shown in Fig. 8A and 8B, Cobb discloses uniform conventions for naming dependent ones of the interrelated components and resources as recited in claims 1, 7, and 10.

Detecting error in individual ones of the interrelated components

Appellant's argues that Cobb fails to disclose detecting error in individual ones of the interrelated components. The Examiner respectfully disagrees.

Cobb discloses the error detection keywords as shown in TABLE 1 (col. 6, lines 62-68 through col. 7, lines 1-37). Noting col. 6, lines 55-58, wherein Cobb discloses "Each symptom string entity has a format `xxxx/vvvvvvvv` where `xxxx` is a keyword that identifies the category of the associated value, `vvvvvvvv`". The following is a keyword that identifies the erroneous device as shown in TABLE 1:
`DEVS`--specifies that `vvvvvvvv` contains a value that identifies a device that is involved in a failure (e.g., **device number, device address**).

Cobb further discloses (col. 6, lines 40-42), "The `PCSS` keyword specifies that the associated value `ssssssss` contains a **unique detection point identifier.**" Therefore Cobb discloses Detecting error in individual ones of the interrelated components as **recited in claims 1, 7, and 10.**

Parsing a log associated with said specific one of the components

Appellant's argues that Cobb fails to disclose parsing a log associated with said specific one of the components. The Examiner respectfully disagrees and would like to point out to Cobb's Fig. 9-10, which shows the steps performed to construct the generic alert from the error log (i.e. *parsing a log*).

Noting col. 8, lines 1-25, Cobb discloses the steps for constructing the generic alert from the error log, "**In the first step**, the generic alert data and **probable causes** sub vectors are built from information in the generic alert descriptor entry 13 selected by

the ALERTDSC keyword on the EDDC process cell 35 (FIG. 7). FIG. 9 illustrates a detailed mapping of the generic alert descriptor entry fields to the generic alert data and probable causes sub vectors. **In the second step**, the generic alert causes sub vector (s) is built from information in the generic alert causes entry (s) selected by the ALERTCSE keyword on the EDDC process call 35. FIG. 9 again illustrates this mapping. **In the third step**, the generic alert recommended action sub vector is built from information in the generic alert recommended action entry (s) selected by the ALERTRAC keyword on the EDDC process call 35. The final three steps in the construction of the generic alert require information from the software problem error log record stored on error log 55." The above steps performed by Cobb's error detection system, reads on the claimed limitation of parsing a log associated with said specific one of the components to determine whether said error condition arose from a fault in one of the interrelated components and resources named in said associated log, as recited in claims 1, 7, and 10.

Therefore, for the reasons stated above, Cobb discloses the claimed invention, as recited in claims 1, 7, and 10.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Elmira Mehrmanesh

Conferees:

Robert Beausoliel

ROBERT BEAUSOLIEL
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Mark Rinehart

MARK H. RINEHART
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100